

Remarks

Entry of the amendments, consideration of the application, as amended, and allowance of all pending claims are respectfully requested. Claims 1-43 are pending.

This application is being filed to continue prosecution of the claims presented herewith. This application is a continuation application of application Serial No. 09/408,916, filed 9/30/99, hereinafter referred to as the parent application. The independent claims are being amended as a clarification and not in acquiescence to any rejection. Support for the amendment can be found throughout the specification (e.g., p. 7, lines 1-22; FIG. 3, p. 14, lines 22-28; p. 20, lines 4-15). Thus, no new matter is added.

In an Office Action, dated September 26, 2003, received for the parent application, claims 1, 2, 5, 6, 9-16, 19, 20, 22-31, 34, 35 & 38-44 were rejected under U.S.C. 102(e) as being anticipated by Gard et al. (U.S. Patent No. 6,463,584); and claims 3, 4, 17, 18, 32 & 33 were rejected under 35 U.S.C. 103(a) as being unpatentable over Gard in view of Khalidi et al. (U.S. Patent No. 5,561,799). Applicants respectfully, but most strenuously, traverse these rejections to any extent deemed applicable to the pending claims.

In one aspect, applicants' invention is directed to updating components of a computing environment in such a way that both the new version of a component associated with a unit of work and the old version of the component associated with that unit of work are concurrently processing the same or different portions of that unit of work. For example, assume there are multiple copies of a program processing a unit of work. Then, assume that one copy is updated to a new version, whereas the other copies remain at the old version, at least temporarily. The new version and the old versions are able to concurrently process one or more portions of the unit of work, since the new version emulates the old versions. That is, the new version mimics behavior of the old versions.

In one embodiment, to update the program, the program is stopped, a new version is loaded and the new program is started. During this process, one or more other copies of the program can continue processing one or more portions of the unit of work. Subsequent to starting the new version of the program, the new version joins the copies of the old version in

processing one or more portions of the unit of work. This technique is distinct from other methodologies in which the running state of the program is transferred from an old version of the program to a new version of the program, as in Gard.

As one particular example, applicants claim a method of updating components in a computing environment (e.g., independent claim 1). The method includes, for instance, updating a component of the computing environment which is associated with at least a portion of a unit of work from one version to another version; and emulating, by the updated component, the one version, while at least one other component of the computing environment associated with the unit of work remains at the one version, wherein the updated component processes at least a portion of the unit of work concurrent to the non-updated component processing at least a portion of the unit of work. Thus, in applicants' claimed invention, an updated version of a component emulates the non-updated version of the component. This allows both components, the updated component and the non-updated component, to concurrently process one or more portions of the unit of work. For instance, assume there are two copies of a program, Program C, and one of those copies is updated to add new functionality, and thus, is referred to as Program C'. As claimed by applicants, Program C' emulates Program C. That is, Program C' imitates (see, e.g., Webster's Ninth New Collegiate Dictionary) Program C allowing Program C' to continue processing the unit of work and to ignore its added functionality for the time being. This is very different from the teachings of Gard.

For example, in Gard, there is no discussion, teaching or suggestion of emulation. In particular, there is no discussion, teaching or suggestion of an updated component associated with a unit of work emulating a non-updated component also associated with that unit of work. There is no imitating by an updated component of a non-updated component in Gard. Instead, Gard specifically teaches that in order to use the new software, data incompatible with the new software is to be converted before the new software can execute. Thus, the new software cannot even run until the data is adapted to the new software.

For example, it is explicitly stated in Col. 2, lines 33-37:

Thus, according to the present invention, the switch over from the old, software to the new software requires that the complete.state

as represented in all data of the old software is copied and, if necessary, simultaneously converted, to the new software.

There is no teaching of emulation in Gard. If Gard was able to emulate the old software, then there would be no need for this conversion in order to execute the new software. The new software would be able to execute anyway.

Support for the rejection of the emulating feature of applicants' claims is indicated in the first Office Action for the parent application in which it is stated that emulation is taught at Col. 5, lines 40-43 and Col. 6, lines 24-29 of Gard. However, a careful reading of those sections merely teaches that before the new software can execute work, data needs to be copied from the old software to the new software and converted, if necessary. There is absolutely no description at all of the new software emulating the old software, as claimed by applicants.

Since Gard fails to teach emulation of the old software, Gard fails to describe, teach or suggest applicants' claimed element of emulating, by said updated component, said one version, while at least one component of said computing environment associated with said unit of work remains at said one version. For this reason alone, applicants respectfully submit that Gard does not anticipate applicants' claimed invention.

In addition to the above, applicants respectfully submit that there is no need for the updated version of the software in Gard to emulate the non-updated version of the software, since the updated version in Gard does not begin executing, until the non-updated version of that software ceases to execute. That is, there is no concurrent execution of the updated version of the software and the non-updated version of that software, as claimed by applicants. At the most, the standby partition housing the updated version receives data and converts the data to ready the updated version for execution, but the updated version of the software does not process units of work, until the non-updated version of that software ceases to process units of work. This is explicitly described throughout Gard.

For example, in Col. 2, lines 33-37, it states:

Thus, according to the present invention, the switch over from the old, software to the new software requires that the complete.state

as represented in all data of the old software is copied and, if necessary, simultaneously converted, to the new software.

Further, in the Abstract, it states:

Data is transferred from the executing partition to the standby partition in a scaleable way and as soon as the same state is achieved for the standby partition and the executing partition the execution is switched to the new, software.

Yet further, in Col. 8, lines 15-19, Gard states:

Significant for the state copying method is that there is never a concurrent execution of software going on in the executing partition and the standby partition except for the update function itself.

Even further, FIG. 3 of Gard explicitly shows that the software that is to be updated (i.e., the software that has an old version and a new version), OPX, only executes in one partition at a time. It does not execute in the standby partition. Standby is known and is defined as being held in reserve ready for use (see, e.g., Webster's Ninth New Collegiate Dictionary). Thus, the standby partition is a partition getting ready to execute, but is not executing the updated software yet. Again, in Gard, the software that is to be updated is only executed in one partition at a time.

To further explain, in FIG. 3, two partitions are shown, an executing partition (EX) and a standby partition (SB). In Steps 1, 2, 3 and 4, it is shown that an application OPX (which is the software to be updated) is running in the executing partition. In Steps 3 and 4, data is copied from the executing partition to the standby partition. In Step 5, once the data is copied, the new software in the standby partition can now execute and the partition that was executing now becomes the standby partition. This is clearly shown in Step 6. As shown in this figure, the software to be updated only executes in the executing partition; it does not execute in the standby partition. This is explicitly admitted in Col. 8, lines 15-18 of Gard, which states: "Significant for the state copying method is that there is **never** a concurrent execution of software going on in the executing partition and the standby partition except for the update function itself." (emphasis added). Thus, in Gard, software that is to be updated is never executing in both partitions. At most, tasks used to prepare the updated software for execution, such as data conversion, may take place, but the updated software is not executing, while the old software is still executing.

Now to address the exception in the above quote. The exception in the above quote is for the update function itself. The update function is a function that facilitates the updating. For instance, it is the function that transfers data between the executing partition and the standby partition to enable the standby partition at some point to become ready to execute. The update function itself is not updated. Thus, there is no concurrent execution of a new version of software (“updated component of the unit of work”) and an old version of the software (“non-updated component of the unit of work”), as claimed by applicants. There is no concurrent execution of components to be updated. The only concurrency is of an update function, which is not updated itself; or of different components that are not processing the same unit of work (e.g., one version of software and a separate function, such as data conversion). Thus, there is no concurrent processing in Gard of an old version of software and a new version of software.

Based on the foregoing, applicants respectfully submit that Gard fails to describe, teach or suggest at least applicants’ claimed element of “emulating by said updated component, said one version, while at least one other component of said computing environment associated with said unit of work remains at said one version.” Further, applicants respectfully submit that Gard fails to describe, teach or suggest “emulating … wherein the updated component processes at least a portion of the unit of work concurrent to the non-updated component processing at least a portion of the unit of work.” Thus, applicants respectfully request an indication of allowability for independent claim 1, as well as the other independent claims.

The dependent claims are patentable for the same reasons as the independent claims, as well as for their own additional features. Khalidi does not overcome the deficiencies of Gard. For all of the above reasons, applicants respectfully request an indication of allowability for all pending claims.

If the Examiner believes any issue could be expeditiously resolved through a telephonic interview, the Examiner is invited to call applicants' undersigned representative.

Respectfully submitted,

Blanche E. Schiller
Blanche E. Schiller
Attorney for Applicants
Registration No.: 35,670

Dated: March 10, 2004.

HESLIN ROTHENBERG FARLEY & MESITI P.C.
5 Columbia Circle
Albany, New York 12203-5160
Telephone: (518) 452-5600
Facsimile: (518) 452-5579